Enhanced Evolutionary Algorithms for Multidisciplinary Design Optimization: a Control Engineering Perspective

Gabriella Dellino[1], Paolo Lino[2], Carlo Meloni*[2] and Alessandro Rizzo[2]

[1] Dipartimento di Matematica - Università di Bari,
Via E. Orabona, 4. I – 70125 Bari – Italy
gdellino@dm.uniba.it
[2] Dipartimento di Elettrotecnica ed Elettronica - Politecnico di Bari,
Via Re David, 200. I – 70125 Bari – Italy
lino@ieee.org, {meloni, rizzo}@deemail.poliba.it
*Corresponding author

Summary. The Chapter deals with the application of hybrid evolutionary methods to design optimization issues in which approximation techniques and model management strategies can be used to guide the decision making process in a multidisciplinary context. An enhanced evolutionary algorithmic scheme devoted to design optimization is proposed, and its use in real applications is illustrated in the framework of the multidisciplinary design optimization (MDO). At this aim, a case study is discussed. It relies to the field of automotive engineering in which the design optimization of a system is carried out considering simultaneously both mechanical and control requirements. The studied system is the regulator of the injection pressure of an innovative Common Rail system for Compressed Natural Gas (CNG) engines, whose engineering design optimization includes several practical and numerical difficulties. To tackle such a situation, a multi-objective optimization formulation of the problem is proposed. The adopted optimization strategy pursues the Pareto-optimality on the basis of fitness functions that capture domain specific design aspects as well as static and dynamic objectives. The computational experiments show the ability of the proposed method for finding a satisfactory set of efficient solutions.

1 Introduction

A successful complex system design requires the harmonization of different objectives and constraints. This design task can be modelled as a constrained optimization problem in the space of the design variables. However, for such optimization, due to its dimensionality, complexity, and costs for analysis, a decomposition approach is recommended to enable concurrent execution of smaller and more tractable issues. Multidisciplinary Design Optimization (MDO) offers effective methods for performing the above optimization so as to resolve the trade-off relations among the various

design criteria at the different system and subsystem levels. In this Chapter, the application of evolutionary algorithmic approaches in MDO from a control engineering perspective is considered. Control design tasks often require to conduct some complex and usually time consuming simulation studies to evaluate functions able to capture and describe the dynamic systems behaviour. Evolutionary algorithms (EAs) offer a way to deal with the constrained multiobjective optimization problems arising in that MDO context. However, these methods are not suitable in cases when some underlying functions evaluations are too expensive. The increasing availability of surrogate models for optimization of complex functions is giving new important roles for EAs. Surrogate models are, in general, constructed over a specific domain using sampling techniques based on design of experiments (DOE) joined with computer simulation techniques and adequate meta-modeling techniques such as response surface methodology (RSM), kriging methods, neural networks (NN), and other approximation methods. Evolutionary design methods may work satisfactorily with the surrogate models, finding optimal solutions or fronts (in the case of multiobjective problems) in a robust manner with acceptable computational costs (even when a large number of functions evaluations occurs). These considerations suggest to adopt multiobjective optimization evolutionary algorithms (MOEAs) in order to tackle the problems arising in the design activities. However, general purpose MOEAs require to be equipped with methods to operate with surrogate models leading to the concept of enhanced evolutionary algorithms.

The Chapter is organized as follows. In Section 2 an overview on Multidisciplinary Design Optimization methodology is presented, the relevant literature is addressed, then the topic is discussed under a control engineering perspective. Section 3 is devoted to illustrate the proposed enhanced evolutionary algorithmic scheme which provides different ways to incorporate the designers domain specific knowledge [19], from interactive actions and simulation-based analysis to surrogate-assisted evolution and model management techniques. Section 4 reports on the application of the proposed scheme to a case study from the automotive field. Finally in Section 5 conclusions are drawn.

2 Multidisciplinary Design Optimization: a Control Engineering Perspective

The increasing demand for improved designs within shorter product development cycle times requires the incorporation of optimization theory, tools and practices developed in the mathematical field into the design process. The formal methodologies which incorporate and coordinate these features are usually referred to as MDO. It has evolved as a new discipline that provides a body of methods and techniques to assist engineers in moving system design closer to optimum. Comprehensive reviews of MDO are given in a number of publications including [1, 21, 41].

The incorporation of optimization methods into engineering design problems is not an easy task and it is still an area of research. The main challenges are related with

the problem size and with the high computational effort required. These two characteristics of engineering problems, joined to the organizational issues related to data sharing and communications needs, contrast with the use of traditional optimization techniques during the optimal design process. At this aim, approximation models have been introduced in the multidisciplinary design methodology and proper model management frameworks have been proposed to guide the optimization of these engineering systems.

Complex design problems in engineering are often characterized by multidisciplinary interactions in which participating disciplines are intrinsically coupled to one another. Designers have recognized the need to decompose such systems into a set of more tractable disciplines. This decomposition is usually based on either the engineering disciplines or the mathematical models representing the system. As a result, the design of such complex systems often involves the work of many experts (engineering teams) in various disciplines, each dependent on the work of other groups and knowing quite little about the analysis and software tools available to the partners. Hence, design working groups cannot work isolated, but they have to work in harmony to achieve a consistent and globally satisfactory design. To obtain a satisfactory output starting from a given nominal (tentative) design it is necessary to invoke an iterative solution to the coupled problem which loops through the feed-backs and feed-forwards until some convergence criteria are met. This iterative solution is often referred to as a system analysis. An iterative approach to the design process overcomes the problem of a lack of interaction between the various subsystems to a certain extent, leading to a certain degree of optimality, although this requires a considerable amount of time, effort and resources.

The use of approximations in representing the design space and evaluating design alternatives is essential for MDO algorithms. Approximations provide information about the system under study to the optimizer without the cost of both a detailed explicit description and a more computationally intensive analysis. The use of approximation techniques provides also a way for the decoupling of disciplines which avoids the constant transfer of information among them during an iterative system analysis. Hence, in an iterative scheme, most of the MDO algorithmic schemes include classical optimization procedures to system approximation, thus providing lower-cost computational models of the objective functions and constraints. These models could be less accurate representations of the actual problem or system; or model approximations which are algebraic representations obtained from samples where information is known (e.g. polynomial response surface approximations and kriging predictions). When a design solution to the approximated problem representation is found, a full system analysis is executed, the approximation model is updated and the process repeated until convergence to a satisfactory design solution of the original problem is obtained. Most MDO algorithmic approaches differ in the way these approximation models are obtained and managed in order to guide to the solution of the original problem. All these aspects call for suitable software tools to provide a practical and robust MDO environment.

The specific software characterization may refer to the key MDO functions that can be recognized as: architectural design, problem formulation, problem solution,

and access to information. This characterization guides to individuate the key technical attributes for a MDO software environment as follows: computer power, task decomposition, sensitivity analysis, human interface and data communication. From a process integration and problem solving point of view, key features of MDO software frameworks include: i) ways for quick and easy linking of analysis tools as well as tools to capture users knowledge; ii) an effective support for distributed network-based modeling and optimization; iii) an access to efficient design of experiments (DOE) tools; iv) an access to an adequate set of optimization algorithms; v) an access to an adequate set of model approximation methods; vi) tools to perform trade-off studies between different design responses.

In the control systems domain, generally engineers face completely designed systems, for which desired performances can be obtained employing a control system. In other cases, the only way to obtain high performances is to early combine the physical system design process and the control system development. Yet, following this approach, some obstacles have to be overcome. In fact, knowledge in fields related to the physical system under study is requested to the control engineer to make it possible to identify, during the system design phase, the elements that greatly affect the controlled system performances. For example, some limitations could derive from hardware performances, sensors features, or microcontrollers computational power. At the same time, such approach opens new perspectives for the control theory influence on other engineering disciplines, thus allowing, as a matter of fact, the solution of multidisciplinary optimization problems [20].

The main idea is to check whether satisfactory solutions to the problem there exist, and then to carry out the cooperation between design and control engineers to test different structural, control and sensors configurations to accomplish the task. This MDO approach can benefit from the control engineers skills during the design process. First of all, mathematical modeling gives important instructions during the physical system design process. Sensitivity analysis, which shows the influence of disturbances and parametric variations on the system performances, gives better results if accomplished in early design steps than during the controller development. The control system robustness, which ensures that specifications are satisfied even in presence of model uncertainties, is suitable only for some classes of uncertainties, that can be obtained through adequate system-control combinations.

The physical and control systems parameters tuning can be performed following different approaches from a control engineering perspective. They rely on the satisfaction of performance indices in the frequency and time domains and need suitable system models or time response testings [9].

Time domain performance indices can be classified in two categories. The first one refers to some characteristic points of the system step response. We have to stress that if it is possible to calculate analytically these quantities for low order linear systems, bringing to closed expressions for the control system parameters, facing with a nonlinear system could make the problem very hard to solve. In this case the numerical solution of the model equations can help to overcome this obstacle. The second category uses integral cost functions taking into account the system dynamics, including for instance the tracking error or the system output. This kind of indices can

be exploited to estimate the distance of a controlled output from a desired behaviour, to evaluate the steady state error level, and disturbance rejection capabilities. Obviously, in this case an estimation of the indices during the tuning phase can be easily obtained through dynamical simulations by integrating the system model equations within the specific time interval.

As already observed, mathematical modeling and dynamical simulation represent a needful tool for designing and optimization of complex nonlinear systems [39, 45]. A typical example is represented by automotive engines. Adjustment of mechanical parameters needs the knowledge of actual working conditions and relative influence of each element composing the system. To this aim it is necessary to evaluate the whole system transient behaviour, thus not focusing the analysis on single components or on steady state operations. In this framework, dynamical models can be suitably used for geometrical optimization of mechanical parts, helping in experimentation to understand results, and for designing controllers [11].

Optimization models based on static equations are not completely suitable for the system design, as the interaction of engines sub-systems under transient evaluation is very different from that evident during steady state evaluation [46]. Tipically, the control task for mechanical systems is done in two steps. The first one regards the optimization of the stationary performances of the engine system for different objectives, the second focuses on the optimal feedforward and feedback control of the dynamic control system [10, 11]. Steady state testing involves measuring and analyz-

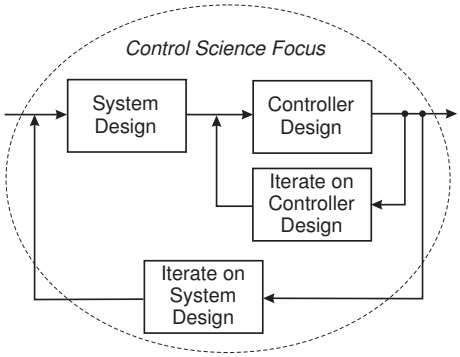

Fig. 1. The integrated design process in a control engineering context

ing engine performance at discrete points within the operating range of the engine. Steady state mapping provides a straightforward route to engine calibration. However, static representation of effects of transient conditions could be an acceptable

approximation only as a first attempt. Summing up, it is difficult to attempt any form of optimization without a detailed knowledge of how the engines sub-components react under dynamic conditions. Steady-state optimization has to be considered, at best, as a baseline for transient investigations. Finally, transient testing is an important part of a systematic engine development program and it is particularly useful in checking the action of the many control algorithms in use on a modern engine. In Fig. 1 the discussed integrated design process is schematically represented.

3 An Enhanced Evolutionary Scheme for Design Optimization

This section describes the algorithmic scheme proposed as a building block of a decision support system in a Computer Aided Engineering (CAE) context playing a relevant role in the MDO iterative multidisciplinary design process.

The proposed system can be classified as a surrogate-assisted design multiobjective optimization tool. This approach results of interest in design activities when some functions to be optimized are computationally expensive calling for an indirect evaluation based on a campaign of experiments or computer simulations. The results of the experimental activity provide a way to construct surrogate models (or meta-models) which are orders of magnitude cheaper to be determined with respect to the exact function evaluations during the optimization process.

The framework has been developed in the MATLAB environment providing integration with MATLAB's Toolboxes and an easy access to almost all domain specific software tools. The description refers first to the general system's implementation scheme, then specific characteristics of the system's elements are discussed. In Fig. 2 a complete scheme of the proposed hybrid framework is depicted. This scheme is composed of five main blocks, namely: 1. Multiobjective Optimizer; 2. Solutions Archive; 3. Meta-model Constructor; 4. Design of Experiments Tool; 5. Solutions Analyzer.

The framework is developed to support the use of different optimization algorithms (e.g. genetic algorithms, particle swarm optimizers or more classic optimization algorithms) and, in its current version, it is equipped with a state-of-the-art evolutionary algorithm as multiobjective optimizer, whose key aspects are discussed in detail in what follows. The framework includes tools for the design and analysis of computer experiments. These tools are devoted to support the construction and the update of the surrogate models on the basis of external computer simulations and other knowledge or experimental activities. The Design of Experiments (DOE) Tool offers a systematic way to study the effects of multiple factors on design results. In other words, the use of DOE provides a suitable method for determining the most significant factors and interactions in a given design task. The access to a rich library of optimization solution methods (in the MATLAB environment) joined with domain specific human expertise (that can appropriately guide the optimization strategy) enables to tackle the solution of the various complex problems faced in design activities. The Meta-model Constructor is based on approximation techniques and provides useful tools to create a simple mathematical model that approximates the

behaviour of an costly computational tool to be used in the optimization process in order to significantly reduce the number of expensive exact analyses. Moreover, these techniques can improve the optimization phase by reducing the effect of noisy phenomena through the use of smooth response surfaces. This approach is at the basis of different response surface representations, used in many MDO applications. In the solution of the trade-off between different design responses, multi-objective optimization procedures often play an important role to individuate efficient candidates and guide the decision making under multiple objectives.

The proposed scheme presents two main methods to deal with the knowledge from the specific domains involved in the project or design development: a) the use of surrogate functions describing either aspects of the system's behaviour (in general using external simulation software tools) or designer preferences, b) the role of the archive of non-dominated design candidates acting as a shared and bi-directional solutions repository in the software framework, and for the overall MDO team.

In what follows each block of the system will be described in detail.

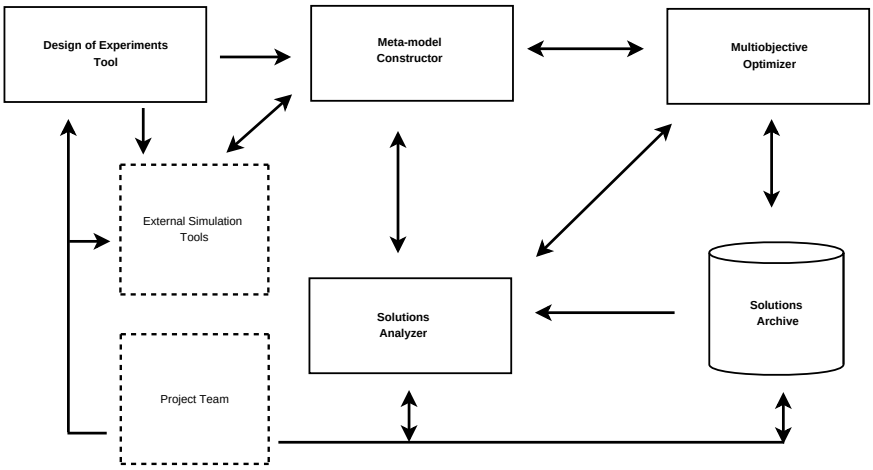

Fig. 2. The design optimization framework

3.1 The Multiobjective Optimizer

The multiobjective optimization algorithm adopted in the current version of the software framework is based on Deb's multiobjective optimizer NSGA-II (Non-Dominated Sorting Genetic Algorithm-II) [4, 7, 35]. We have integrated it in the

MATLAB's Genetic Algorithm and Direct Search Toolbox [27], with clear advantage to access to the utilities given by the MATLAB environment [26, 27, 28, 29]. The resulting algorithm is able to solve constrained optimization problems involving multiple and often conflicting objectives.

The behaviour of the adopted NSGA-II algorithm is sketched in Figs. 3–4 and can be briefly described as follows. At first, a random population is created. Then, starting from the current population P_i, of size M, a new population P_{i+1} is produced through the following steps. The values of the fitness functions for each individual of the population are first computed. Then a selection scheme is applied to form the parent population. At this aim, different selection functions are available to the user, the most popular being uniform, roulette and tournament selection [27]. A child population O_i is created using both crossover and mutation operators. Crossover functions can be chosen by the user among a set including the well known single-point, two-point, and scattered crossover (a random binary vector is created and genes are selected from the first or the second parent, according to the value of each bit) [7, 27, 31]. Mutation can be performed in two possible ways: gaussian and uniform [27]. Elitism is also ensured, thus avoiding the loss of some good solutions eventually found. Crossover and mutation probabilities can be fixed by the user. They are expressed as a fraction of individuals, other than elite children, generated through crossover and mutation, respectively. Thereafter the combined population C_i, of size $2M$, – obtained by joining the parent and offspring populations together – is ranked on the basis of the Pareto domination concept [8], according to the scheme proposed by Deb in [7]. In particular, several non-dominated levels are determined such that non-dominated individuals are assigned a ranking value equal to 1, thus forming the first front. The domination analysis is performed once more, just taking account of the individuals not included in the first front; non-dominated individuals in this set constitute the next front, with a ranking value of 2. The process goes on, by assigning higher and higher ranking values to the next fronts, until every element in the current population has been assigned to a specific non-dominated front. Once all the population has been ranked, the new population P_{i+1} is generated by keeping M solutions from the non-dominated fronts, according to the rank assigned to the individuals in the previous step. If two or more solutions are non-dominated with respect to one another, having the same rank, another comparison is performed, estimating the density of solutions in the neighborhood of each of them. In particular, a crowding distance value is assigned to the individuals belonging to the same front, calculated as follows: for each objective function, the solutions in each front are sorted in ascending order of magnitude. Then an infinite distance value is assigned to solutions having the smallest and highest objective function values; for solutions other than the boundary ones, the distance value is given by (1):

$$d_k = d_k + \frac{f_m(k+1) - f_m(k-1)}{f_m^{max} - f_m^{min}} \,, \tag{1}$$

where d_k is the crowding distance associated to the k-th elements, whose initial value is set to 0, $f_m(\cdot)$ is the value of the m-th objective function of the element specified in brackets, $(k+1)$ and $(k-1)$ are the successor and the predecessor of the k-th individ-

ual. According to the operated sorting, f_m^{max} and f_m^{min} are the maximum and minimum value of the m-th objective function. The calculation of d_k is performed with all the objective functions, finally obtaining the overall crowding distance for each individual. Thus solutions located in lesser crowded regions are preferred. These steps are performed in every generation, till one of some stopping criteria is met [7]. At this aim, the user has to specify his preference using (possibly composing) the following options: maximum number of iterations to be performed, maximum time the algorithm has to run, fitness threshold to be achieved, number of generations or time interval for which there is no improvement in the non-dominated front.

A more detailed exposition on NSGA-II is available in the literature [7].

Optimization Algorithm

begin
  for each generation i
    Parents selection in the current population, P_i
    Determine the offspring population O_i through crossover and mutation
    Join the parents and offspring population, $C_i = P_i \cup O_i$
    Domination analysis of C_i
    Generation of the new population, P_{i+1}
end

Fig. 3. The NSGA-II main loop

Domination Analysis

Input: population P;
begin
  for each pair of individuals (j,k) in P, $j \neq k$
    if j and k are feasible
    j and k are ranked based on the non-domination
    else if j is feasible and k is not
      j dominates k
    else /* j and k are both infeasible */
      j and k are ranked based on the constraints violation
end

Fig. 4. The Domination Analysis Scheme

The algorithm proposed in this work is adapted to deal with constraints by penalizing infeasible solutions: a penalty factor is determined on the basis of both the number of violated constraints, and the distance from the feasible region, which is

determined by taking into account constraints violation as a whole. This means that, for a given solution, each violated constraint causes the corresponding penalty factor to be increased; so, if a solution violates several constraints, it will be more penalized than one violating a smaller number of constraints. Moreover, the more a constraint is violated by a solution, the greater the associated penalty factor will be. However, a greater weight to some constraints can be assigned by the user, e.g. when violation could have dangerous consequences or lead to a system that cannot work. It makes more difficult for these infeasible solutions to survive in the next generations [31, 38]. In fact, constraints violation influences the ranking of the population so that any feasible solution has a better rank than any infeasible one. While all feasible solutions are ranked according to their objective values, infeasible solutions are ranked on the basis of the amount of constraints violation. A different approach is followed for handling box constraints on the decision variables: in fact, if a solution violates this type of constraint, being a variable lower than its lower bound or greater than its upper bound, it is saturated to the corresponding boundary value, i.e. the lower or upper bound respectively, forcing the solution to be feasible.

The algorithm optionally performs the seeding operation in the initial population, i.e. it is possible to specify some individuals to be included into the starting population, letting the others be randomly generated. Initial seeding could be a useful operation when there are some known (or proposed) promising solutions (e.g. the variables values related to some tentative design) – coming for example from a previous design process iteration – and the algorithm can start from these points. Besides the seeding operation, an injection operation of external elements in the current population can be optionally performed during the algorithm's evolution. This operation provides a way to take into account good design candidates (possibly) emerging during the multidisciplinary design activity.

3.2 The Solutions Archive

The proposed framework uses a data structure called Solutions Archive collecting the non-dominated solutions as the result of the multidisciplinary design effort and, in particular, those found during the multiobjective optimization search.

The elements of the archive can be (optionally) involved in the multiobjective optimization process also as candidates to be injected in the current population. Since a solution is classified as non-dominated depending on the context in which it is evaluated, it is clearly necessary to update this archive, removing all dominated solutions (if any). How often the archive is updated, the conditions and the amount of elements to involve in the injection operations are set by the user before the algorithm execution.

Indeed, employing the Solutions Archive reduces the risk of losing desirable (i.e. non-dominated) solutions, since there are no guarantees – in the NSGA-II algorithmic scheme [7] – for them to remain in the current population until the algorithm stops. Moreover, the Solution Archive plays an important role storing best solutions when, facing the characteristic random behaviour of the employed procedures, users

usually organize repetitive optimization campaigns on each instance of their problems.

When requested, injection operations take place removing solutions with lower values of ranking from the current population and substituting them with the oldest non-dominated individuals contained in the archive, that is, individuals that entered the archive sooner (even if different criteria can be adopted).

The archive structure A has a maximum size S (i.e. it may contain up to S elements), but at any iteration it may contain a number $s \leq S$ of members. The correct use of the archive requires a capacity control, a domination and crowding analysis for the elements that are proposed to be collected. In particular, the evolutionary optimizer proposes to the archive the new non-dominated elements contained in its current population. In Fig. 5 the acceptance rule of new elements (i.e. coming from the optimization process as well as from external tools or knowledge) in the archive is illustrated.

Archive acceptance rule
Input: archive A, archive size S, candidate element e;

begin
  if a e dominates any element in A **then**
    delete all dominated elements of A and include e in A
  else if no element of A dominates e and $|A| < S$ **then**
    include e in A
  else if no element of A dominates e and $|A| = S$ **then**
    if the crowding distance of e is better than that of $x \in A$ **then**
      substitute x with e in A
end

Fig. 5. The acceptance criteria of new elements in the archive

3.3 The Solutions Analyzer

The framework provides tools for the analysis of the set of non-dominated elements contained in the Solutions Archive. These tools consist of metrics to evaluate the quality of the obtained optimization solutions and a cluster analyzer.

The set of metrics implemented is based on the state-of-the-art in the field [36, 44, 47] and enables the user to conduct an accurate post-optimality analysis or to organize the dynamic of the optimization process taking into account the on-line evaluation of some indices of quality. To measure the front improvements the Solution Analyzer adopt the generational distance (GD) index, introduced by Van Veldhuizen and Lamont [44], which was originally proposed to measure the distance from a reference front. This index is computed as indicated in (2):

$$GD = \frac{\sqrt{\sum_{i=1}^{n} {d_i}^2}}{n}\,, \tag{2}$$

where n is the number of non-dominated solutions in the given front and d_i is the distance in the objective space (i.e. the space having the objective indices as coordinate axes) between the i-th solution and the nearest solution in the reference front. The user can consider a threshold on the value of GD between the current front (considered as a reference) and the front at a previous iteration as triggering condition for the algorithm termination, starting the analysis of the current results or some local search procedure.

The analysis of results can be carried out also considering other metrics for MOEAs from recent literature [47]. The first metric is the two set coverage (C), which estimates the relative coverage of two fronts obtained by different algorithms or different optimization phases. Given two sets of solutions $X', X'' \subseteq X$, the metric C maps the ordered pair (X', X'') into the interval $[0,1]$ as follows:

$$C = (X', X'') = \frac{|\{a'' \in X''; \exists a' \in X' : a' \succeq a''\}|}{|\{X''\}|}\,; \tag{3}$$

where the notation $a \succeq b$ indicates that a covers b, i.e. $a \succeq b \Leftrightarrow a \succ b$ or $f(a) = f(b)$, while $a \succ b$ indicates that a dominates b, and $f(a)$ and $f(b)$ represent fitness vectors associated to a and b, respectively. In other words, a solution a covers a solution b if and only if either a dominates b or a and b have the same fitness vector. Due to this definition, it can be noted that the function C is not symmetric. The overall coverage obtained with the metric C provides a clear indication of the relative quality of the compared solution sets. The advantage of this metric is that it is easy to calculate, and provides an effective indication of the relative quality of the compared sets. Another metric often referred in the literature is the spacing (S) as a way of measuring the range (distance) variance of neighbouring solutions in the known Pareto front. The Solution Analyzer provides this metric according to the definition reported in [44]:

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d} - d_i)^2}\,; \tag{4}$$

where

$$d_i = min_j(|f_1^i(x) - f_1^j(x)| + |f_2^i(x) - f_2^j(x)| + \ldots + |f_m^i(x) - f_m^j(x)|)\,;$$

and $\bar{d} = average(d_i)$, $i, j = 1, \ldots, n$, while n is the number of elements in the considered set of solutions, m is the number of fitness functions, and f_k^h indicates the k-th component of the fitness vector associated to the h-th element in the set. The metric S provides a measure of the spread of points throughout the considered Pareto-front. The lower is the value for this metric, the more uniformly spaced will be the points in the front. In particular, a value of zero for S indicates that all members of the front are equidistantly spaced. In further developments of the Solutions Analyzer, metrics based on some estimation (e.g. based on ideal and nadir points) of the Pareto front, such as the Hyperarea Ratio [36], will be also considered.

The cluster analyzer is based on a method recently proposed by Prieto and Peña [40]. This method uses a procedure to identify clusters in multivariate data using information obtained from the univariate projections of the sample data onto certain directions. The directions are chosen as those that minimize and maximize the kurtosis coefficient of the projected data. This tool could be used to analyze solutions in the space of the fitness functions as well as in the space of the variables. The goal of the cluster analysis could be the individuation of promising search regions to explore applying some local search procedures or to improve the meta-models in use. Other uses of the cluster analysis are devoted to recognize, under the supervision of the experts, groups of similar and close solutions : i) in the space of the variables, in order to accomplish a more robust design; ii) in the space of the fitness functions (or with respect to some specific index of quality), in order to conduct an isoperformance analysis [6].

3.4 Enhancing the Algorithm with Meta-Models

A common engineering practice is the use of approximation models in place of expensive computer simulations to drive a multidisciplinary design process based on nonlinear and often multiobjective programming techniques. The use of approximation strategies is designed to reduce the number of detailed and costly computer simulations required during optimization while maintaining the pertinent features of the design problem. Model management strategies coordinate the interaction between the optimization and the fidelity of the approximation models so as to ensure that the process converges to a satisfactory solution of the original design problem. Thus, approximations may play an important role in MDO by offering system behaviour information at a relatively low cost.

The framework is developed with the aim to use different specific approximation techniques. These techniques mainly include Kriging Models, Artificial Neural Networks and Response Surface Methodology. This section briefly introduces the kriging technique currently adopted in the framework and used in the case study considered in this Chapter. The obtained approximation models can be directly used in fitness evaluations in order to reduce the overall cost of fitness calculations or domain specific analysis of the system. This practice may reduce the cost of fitness evaluations most significantly. However, the application of meta-models to evolutionary computation is not straightforward and there are two major questions in using them for the fitness evaluation: i) ensure that the evolutionary algorithm converges to the global optimum or to a near-optimum of the original fitness function; ii) reduce as much as possible the total computational cost.

Another essential point is represented by the difficulty to construct a meta-model that is globally correct due to the high dimensionality, ill distribution and limited number of experimental samples. It is found that if an approximated model is used for fitness evaluation, it is very likely that the evolutionary algorithm will converge to a false optimum, i.e. an optimum solution of the approximation model, which is not optimum with respect to the original fitness function. Therefore, it is essential in most cases that the approximation model should be used together with the original fitness

function in order to update and improve the meta-model during the optimization process. This conducts to the important issue called model management or evolution control [14] that will be discussed in what follows.

Surrogate models update strategies have widely been studied in the literature [4, 13, 14, 16, 17, 18, 35, 37]. The basic idea is to select the location of the next sampling data in such a way some criteria are satisfied. These can be a measure of information gain or approximation error reduction. It has been shown that active data selection can improve the generalization ability of surrogates without increasing the number of data samples. Active data selection is also important in the case that the sample data have been collected from an optimization process and therefore, the issue turns in how to select a subset of the data for an efficient surrogate improvement.

The algorithm implementation supports the use of successive approximation models of fitness functions and enables different model management approaches [14, 35]. In Fig. 6 a schematic description of the whole optimization process is reported.

The Meta-Model Construction and the Design of Experiments Tool

In the past, design mainly consisted of experimentation and prototyping phases. Nowadays, powerful computer simulations, are widely available for engineering design and analysis in the Computer Aided Engineering (CAE) context. Examples of CAE tools can be found in almost all engineering domains enabling the designers to simulate the performance of a system [43]. However designers are still confronted with the problem of finding settings for a, often large, number of design parameters [12]. These parameters should be set optimal with respect to several simulated system characteristics. These characteristics, called response parameters, may originate from different engineering disciplines. Since there are many possible design parameter settings and computer simulations are time consuming in many cases, the key question becomes how to find the best possible setting with a minimum number of costly computer simulations.

To deal with these issues, the proposed framework include two blocks devoted to the design of simulation experiments and the construction of the meta-models, respectively. We consider first the case in which a set of experimental data is available, (i.e. from a previous campaign of computer simulations) and an approximation model, able to adequately represent non-simulated configurations, is requested. Among the methods proposed in the literature, we consider three classes of approximations: kriging methods, neural networks and response surface methodology.

A kriging model can be seen as a combination of a global model plus a localized approximation, as reported in (5):

$$y(\mathbf{x}) = f(\mathbf{x}) + Z(\mathbf{x}) \,, \tag{5}$$

where $f(\mathbf{x})$ is a known function of $\mathbf{x}$ as a global model of the original function, and $Z(\mathbf{x})$ is a Gaussian random function with zero mean and non-zero variance that represents a local deviation from the global model. Usually, the regression model $f(\mathbf{x})$ appearing in (5) can be written as

$$f(\mathbf{x}) = \sum_{i=1}^{p} \beta_i f_i(\mathbf{x}) \,, \tag{6}$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, p$, are polynomial terms (typically of order 1 or 2) and, in many cases, they are reduced to constants. The coefficients β_i, $i = 1, \ldots, p$, are regression parameters. The covariance of $Z(\mathbf{x})$ is expressed by (7):

$$Cov[Z(\mathbf{x}^{(j)}), Z(\mathbf{x}^{(k)})] = \sigma^2 \mathbf{R}[R(\theta, \mathbf{x}^{(j)}, \mathbf{x}^{(k)})], \qquad j, k = 1, \ldots, N_s \,, \tag{7}$$

where N_s is the number of sample points, σ^2 is the so called process variance, $\mathbf{R}$ is the correlation matrix and $R(\theta, \mathbf{x}^{(j)}, \mathbf{x}^{(k)})$ is the correlation function between any two of the N_s samples $\mathbf{x}^{(j)}$ and $\mathbf{x}^{(k)}$, with unknown parameters θ. $\mathbf{R}$ is a symmetric matrix of dimension $N_s \times N_s$, with diagonal elements equal to 1; the form of the correlation function $R(\theta, \mathbf{x}^{(j)}, \mathbf{x}^{(k)})$ can be chosen by the user on a variety of correlation functions proposed in the literature, even if the Gaussian correlation function is used most frequently.

The construction of the kriging model is based on an estimation of the unknown parameters by maximizing a likelihood function using numerical optimization techniques to determine the vector θ of the correlation parameters used to fit the model [25]. In the implemented system, the user has to indicate a starting value $\theta = \theta_0$ and two bounds on the values of its elements (i.e. a lower bound *lowbd* and an upper bound *upbd*, respectively). One advantage of using kriging models is that a confidence interval of the estimation can be obtained without much additional computational cost. Note, however, that it is necessary to perform matrix inversions for estimating the output in the kriging model, which may require high computational costs if the size of the approximation problem increases.

In the proposed framework, kriging models are constructed using algorithms and functions of the MATLAB's Toolbox DACE [25]. In order to build a starting set of elements at the basis of the approximation model, several data sampling methods have been suggested in the field of design of experiments. The DACE toolbox offers a limited support to this preliminary activity, including the rectangular grid – obtained considering all possible combinations of levels for the factors involved in the experiments – and the Latin Hypercube Sampling: this strategy assumes a uniform distribution of the possible levels for each factor and samples the hypercube in the design region in such a way that selected points never form a pair on the same hyperplane. To provide other methods among which the user can choose the most appropriate one, depending on the particular problem he is tackling, the proposed framework integrates also some specific utilities from the MATLAB's Statistics Toolbox [29]: the implemented functions provide tools for full and fractional factorial designs, response surface and D-optimal designs [33, 34].

In addition to Kriging Models, other approximation techniques often used in an MDO environment are Neural Networks (NN) and Response Surface Methodology (RSM). The first technique has gained a significant amount of attention for the mapping of functions and, in the MDO context, much of this work was on using the response surface approximations to replace complex discipline analyses. Response Surface Methodology (RSM) is a polynomial approximation based on the

least squares technique. It has been applied by a number of researchers in some complex engineering problems. When creating RSM models, it is possible to identify the significance of different design factors directly from the coefficients in the normalized model. For problems with a large dimensionality, it is important to use linear or second-order models to narrow the range of design variables to the most critical ones to reduce the dimensionality in approximation. In optimization, the smoothing capability of RSM allows quick convergence of noisy functions. Although these techniques have different advantages, they also have some disadvantages. Since they can be, in some contexts, competitive with the Kriging Method [15], they are valid candidates to be included in the proposed framework as alternative to it. In particular, the developed software tool has an easy access to both of them as utilities recently included in the MATLAB's Neural Networks Toolbox and Statistics Toolbox, respectively [26, 29].

Model Update Strategies

The model management needs that, in evolutionary computation using approximated models, the original fitness function is used to evaluate some of the individuals or all individuals in some generations [14]. These evaluations are used to update the metamodel during the optimization process with the hope to obtain an accurate approximation in the search regions under exploration by the algorithm. Clearly, according to this procedure, in the multiobjective optimizer the fitness functions based on approximated models may change during the algorithm's evolution.

The model management in the proposed framework can be addressed in three main approaches from the evolution control point of view: i) the approximation model can be assumed to be of high-fidelity and therefore, the original fitness function is not used at all in the evolutionary computation process; ii) the individual-based evolution control; and iii) the generation-based evolution control.

In the individual-based control, in each generation, some of the individuals are evaluated with the approximation model and others using the original function. In individual-based evolution control, different strategies (e.g. either a random strategy or a best strategy) can be used to select the individuals to be controlled. In the best strategy, the best individual (based on the ranking evaluated by the approximation model) in the current generation is re-evaluated using the original function, while in the random strategy, the individuals to be evaluated are selected randomly. Individual-based evolution control can be carried out only in a selected number of generations according to a specific parameter to be set by the user.

Generation-based evolution control has also been implemented. Generation-based evolution control can be carried out when the evolutionary algorithm converges on the approximation model. In a more practical way, evolution control can be carried out once in a fixed number f_N of generations. As default option, we adopt this simple approach basing on the elements contained in the archive of non-dominated elements.

The proposed framework is designed to support different active learning strategies based on the analysis of the non-dominated elements contained in the archive.

The analysis can be either a plain automatic procedure or an interactive procedure incorporating experts considerations during the design optimization process. Since the archive contains the best front obtained so far by the optimization procedure, its elements are valid candidates to be used as new samples to update the surrogate on the basis of their evaluation with a more detailed and expensive simulation model. A first (and straightforward) approach uses all the elements of the archive for surrogate update. The frequency of the update operation and the number of elements to consider can be fixed by the user before the algorithm's execution, or they can be alternatively based on the dynamic of the optimization process and on some measures of the approximation error on the elements of the archive. The latter solution will be included in further developments of the software. Other approaches are based on the analysis of the archive conducted automatically or under the supervision of an expert. This analysis may be conducted both in the domain of the fitness functions and in that of the variables in order to individuate regions in which intensify the sampling.

4 Case Study: Optimal Design of a Pressure Controller of a CNG Injection System

Mechanical systems represent a typical example where the design process has to be optimized to achieve good performances. In mechanical systems there are many complex optimization design problems that come multi-objective in nature. Recent restrictive regulations concerning internal combustion engines have encouraged automotive companies and researchers to propose innovative solutions, developing environmentally friendly vehicles still guaranteeing good performances. In particular, the injection system is a critical component for the reduction of pollutant emissions, noise and fuel consumption [2, 22, 24]. In fact, to achieve this goal it is necessary to exactly meter the injected fuel by controlling both the injection pressure and the injectors opening time. Injection dynamics is greatly affected by the components and control system design process. In this Chapter we consider a Compressed Natural Gas injection system as a case study to apply the proposed enhanced evolutionary approach, taking into account both mechanical and control objectives and constraints.

The present section describes the system under study. Firstly, we introduce the Compressed Natural Gas (CNG) injection system and its components, as presented in [2] and [23], focusing on the pressure controller features. Since our aim is to clarify the reasons leading to the development of the optimization model, the injection system functioning is briefly described. Then, the multidisciplinary design optimization issues are discussed. Finally, the application of the evolutionary optimizer to the case study is described and the computational results are illustrated.

4.1 The CNG Injection System and the Injection Pressure Controller

We consider a system composed of the following elements (Fig. 7): a fuel tank, storing high pressure gas, a pressure controller, driven by a solenoid valve, and the fuel

Optimization Process
Input: design specification;
Output: archive of non-dominated elements, reason of termination, last population generated;

Initialization:
Parameters to be set:
f_{upd} – update frequency of the archive
n_{inj} – number of elements to be injected from the archive
f_{inj} – frequency of injection operations
f_N – update frequency of the surrogate model
Acquire design proposals (if any)
Generate a set of experimental data through DOE techniques
Invoke external simulation tools
Build the surrogate model
Generate a random population for the evolutionary algorithm

begin
if seeding required **then**
Load proposed individuals into the initial population
Evaluate all individuals in the initial population
Domination analysis and ranking of the initial population
while any of the stopping criteria is not satisfied
Create a new population according to the adopted evolutionary algorithm
Evaluate all individuals in the population
for each non-dominated individual e in the population
Archive acceptance rule(e)
if mod (generation, f_{upd}) = 0 **then**
Update the archive, removing dominated elements
if mod (generation, f_{inj}) = 0 **then**
Replace the worst n_{inj} dominated elements from the current population with the oldest n_{inj} individuals in the archive
Domination analysis and ranking of the updated population
if mod (generation, f_N) = 0 **then**
Update the surrogate model
end

Fig. 6. The Optimization Process Scheme

metering system, consisting of a common rail and four electro-injectors. The controller reduces the pressure of the fuel supplied by the tank, and sends it to the common rail, feeding the electronically controlled injectors. Then the injectors send the gas to the intake manifolds to obtain the proper air/fuel mixture. The injection flow only depends on rail pressure, which is almost equal to the main chamber pressure, and injection timings, which are precisely driven by the Electronic Control Unit.

The pressure controller includes a main chamber, whose volume and inlet section depend on the axial displacement of a piston integral with a spherical shutter, and

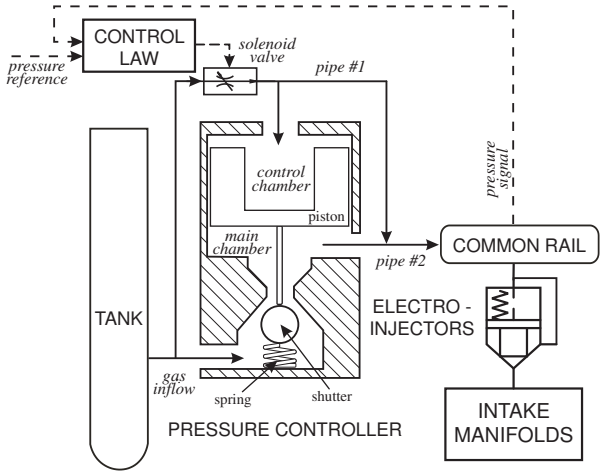

Fig. 7. Block Scheme of the CNG Common Rail injection system

a control chamber, whose pressure is regulated by the solenoid valve. Piston and shutter dynamics are determined by the equilibrium of the applied forces, which are the pressure forces due to the fluid stored in the control chamber, in the main chamber and in the tank, the flow drag force, and the elastic force of a preloaded spring, which tends to get the shutter closed.

When the solenoid valve is energized, the fuel enters the control chamber, causing the pressure on the upper surface of the piston to build up. As a consequence, the piston is pushed down with the shutter, letting more fuel to enter in the main chamber, where the pressure increases. On the contrary, when the solenoid valve is non-energized, the pressure on the upper side of the piston decreases, causing the piston to raise and the main chamber shutter to close under the action of the preloaded spring (see [2] and [23] for further details).

4.2 Mechanical and Control Design Optimization Issues

In this case study, we are interested in the design optimization of the pressure controller, focusing on both mechanical and control issues. This approach allows a wider analysis of the optimization problem, thus putting it in a MDO framework.

From a mechanical point of view, to improve efficiency and shorten the actuation delay, it is necessary to reduce the friction force between the piston and the lateral surface of the pressure controller. At the same time, the accurate control of the injection pressure requires the minimization of the gas leaks between the control and

Table 1. Problem Formulation: Decision Variables

Symbol	Description	Unit
D_i	piston internal diameter	(mm)
h_i	piston internal height	(mm)
A_1	pipe ♯1 section	(mm^2)
V_{rail}	rail volume	(cm^3)
d_0	shutter diameter	(mm)
r_p	piston external radius	(mm)
h_p	piston external height	(mm)
r_{cyl}	cylinder height	(mm)

the main chamber of the pressure controller, due to the pressure gradient across the piston surfaces.

Besides these mechanical objectives, we can include some control specifications in the optimization process, regarding system response and disturbance rejection. To this end, we have to analyze the main chamber, the control chamber and the common rail filling and emptying behaviours, and the piston dynamics. The former are approximatively described by a first order dynamics. Considering the forces involved, the latter can be represented by a second order model, taking in mind that good performances require an overdamped step response with low rise time.

The optimization model developed to achieve the aforementioned objectives is described in what follows.

4.3 Problem Formulation

The decision variables to be optimized are listed in Table 1. The optimization problem can be expressed in the following form:

$$\min_{\mathbf{x}\in X} \left\{ \mathbf{f}(\mathbf{x}) \in \mathbb{R}^7 \middle| \mathbf{g}(\mathbf{x}) \geq \mathbf{0}, \mathbf{h}(\mathbf{x}) \geq \mathbf{0} \right\} , \tag{8}$$

where $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}), f_4(\mathbf{x}), f_5(\mathbf{x}), f_6(\mathbf{x}), f_7(\mathbf{x})]$ is the objective functions vector, whose components are defined in Table 2, and $\mathbf{x} \in \mathbb{R}^8$ is the vector of decision variables; $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_{10}(\mathbf{x})]$ and $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \ldots, h_{16}(\mathbf{x})]$ are the vectors of constraints, whose components are defined in Table 3.

Parameters and constants employed in the expressions of objective functions and constraints are listed in Tables 4–5. Objective functions and constraints are obtained through physical considerations [23, 42, 48] and by taking into account the targets expressed in the Section 4.2, as explained in the following.

The friction force between the piston and the lateral internal surface of the pressure controller is expressed by $f_1(\mathbf{x})$, while the gas leaks reduction is modeled through the objective function $f_2(\mathbf{x})$.

Table 2. Problem Formulation: Objective Functions

Obj.	**Expression**	**Unit**
$f_1(\mathbf{x})$	$k_1 \left[\frac{r_{cyl}^2 - r_p^2}{\ln(r_{cyl}/r_p)} - 2r_p^2 \right] + k_2 \frac{h_p}{\ln(r_{cyl}/r_p)}$	(N)
$f_2(\mathbf{x})$	$k_3 \left[\frac{r_{cyl}^2 - r_p^2}{\ln(r_{cyl}/r_p)} - 2r_p^2 \right] - \frac{k_4}{h_p} \left[r_{cyl}^4 - r_p^4 - \frac{(r_{cyl}^2 - r_p^2)^2}{\ln(r_{cyl}/r_p)} \right]$	(m^3/s)
$f_3(\mathbf{x})$	$\frac{V_{1,pipe} + \frac{\pi}{4} D_i^2 h_i}{A_1}$	(m)
$f_4(\mathbf{x})$	$\frac{V_{mc} + V_{2,pipe} + V_{rail}}{\pi d_0 \cos\beta_s \sin\beta_s H_{max}}$	(m)
$f_5(\mathbf{x})$	$\left\lvert \frac{a\, \bar{p}_{mc}^2}{b_2\, d_0^3\, \bar{p}_{mc}\, \bar{p}_t - r_p^2\, d_0\, b_1 (\bar{p}_{cc} - \bar{p}_{mc}) \bar{p}_t - a\, \bar{p}_{mc}\, \overline{ET}} \right\rvert$	(Pa)
$f_6(\mathbf{x})$	$\frac{\pi}{4} \frac{\rho_{steel}}{k_s} \left(4r_p^2 h_p - D_i^2 h_i \right)$	(ms^2)
$f_7(\mathbf{x})$	$\begin{cases} \frac{2M}{\sqrt{4k_s M - b^2}} \left[\pi - \arctan \frac{\sqrt{4k_s M - b^2}}{b} \right], & M - \frac{b^2}{4k_s} > 0 \\ \left\lvert \frac{2M}{-b + \sqrt{b^2 - 4k_s M}} \right\rvert, & M - \frac{b^2}{4k_s} \leq 0 \end{cases}$	(ms)

To evaluate the rail filling dynamics, we suppose the injectors closed and the main chamber inlet section opened, thus obtaining the maximum gas inflow. Then we try to minimize the time constant of the control chamber, which is expressed by $f_3(\mathbf{x})$, unless a multiplicative constant that has been neglected. Analogous considerations can be done regarding the main chamber and the rail, leading to the objective function $f_4(\mathbf{x})$.

To express the system disturbance rejection capabilities, we consider a second order linear model of the injection system [23]. By keeping the tank pressure and the valve driving signal constant, we apply step variations of the injectors driving signal, then we evaluate the rail pressure changes in steady state conditions, obtaining the objective function $f_5(\mathbf{x})$.

As for the piston dynamics, we have searched for solutions of the optimization problem that could guarantee an overdamped response and a low rise time. The corresponding objectives are given by the functions $f_6(\mathbf{x})$ and $f_7(\mathbf{x})$. The former is the square of the reciprocal of the natural frequency, the latter represents the system time constant or rise time for underdamped and overdamped systems, respectively. In the expression of f_7, M and b are given by (9–10):

$$M = \frac{\pi}{4} \rho_{steel} \left(4r_p^2 h_p - D_i^2 h_i \right) , \tag{9}$$

$$b = 0.5 C_D (\pi r_p^2) \rho_{gas} + \frac{2\pi \mu_{oil} h_p}{ln(r_{cyl}/r_p)} . \tag{10}$$

Decision variables in Table 1 need to satisfy specific conditions to represent feasible solutions of the optimization problem, thus restricting the search space. First

Table 3. Problem Formulation: Constraints

Constraint	**Expression**
$g_1(\mathbf{x})$	$2\,\frac{\alpha_r}{100}\,r_p - D_i$
$g_2(\mathbf{x})$	$\frac{\alpha_h}{100}\,h_p - h_i$
$g_3(\mathbf{x})$	$d_0 - d_s$
$g_4(\mathbf{x})$	$r_{cyl} - r_p - \varepsilon_p$
$g_5(\mathbf{x})$	$\left\|k_5\,r_p^2 + k_6\right\| - 10\left\|k_7\,(4r_p^2\,h_p - D_i^2\,h_i)\right\|$
$g_6(\mathbf{x})$	$\frac{b}{2\sqrt{k_s M}} - \bar{\delta}$
$g_7(\mathbf{x})$	$2\,r_p - D_i - (1 - \frac{\alpha_x}{100})(2r_{p,proj} - D_{i,proj})$
$g_8(\mathbf{x})$	$h_p - h_i - (1 - \frac{\alpha_x}{100})(h_{p,proj} - h_{i,proj})$
$g_9(\mathbf{x})$	$(1 + \frac{\alpha_x}{100})(2r_{p,proj} - D_{i,proj}) - (2\,r_p - D_i)$
$g_{10}(\mathbf{x})$	$(1 + \frac{\alpha_x}{100})(h_{p,proj} - h_{i,proj}) - (h_p - h_i)$
$h_i(\mathbf{x})$	$x_i - (1 - \frac{\alpha_x}{100})\,x_{i,proj}, i = 1,\ldots,8$
$h_{i+8}(\mathbf{x})$	$(1 + \frac{\alpha_x}{100})\,x_{i,proj} - x_i, i = 1,\ldots,8$

of all, we impose the non-negativity constraint on each decision variable, as they represent physical quantities for which a negative value has no significance.

Other constraints derive from geometrical relations among the variables. In particular, $g_1(\mathbf{x})$ and $g_2(\mathbf{x})$ express that the control chamber and piston sizes (in terms of diameters and heights) are related by two constants, α_r and α_h, set through experimental tests. To let the pressure controller work properly, the shutter diameter must be greater than the seat diameter, as imposed by constraint $g_3(\mathbf{x})$. The cylinder and piston radiuses are related to the clearance between cylinder and piston, as stated in $g_4(\mathbf{x})$.

We can introduce another constraint aiming to improve the actuation speed. More in detail, it is affected by the shutter and piston inertia, which has to be negligible in comparison to the other hydraulic forces involved. In order to keep this hypothesis valid, some decision variables are subjected to the constraint $g_5(\mathbf{x})$.

Further, we consider that a piston overdamped dynamics would be desirable, but as it is not always possible to achieve such a requirement, we impose a damping ratio greater than $\bar{\delta}$, whose value is reasonably chosen to guarantee a reduced oscillatory behaviour in the system response. Thus the constraint $g_6(\mathbf{x})$ is obtained.

Table 4. Problem Formulation: Parameters

Symbol	**Description**	**Unit**
$V_{1,pipe}$	pipe ♯1 overall volume	(m^3)
$V_{2,pipe}$	pipe ♯2 overall volume	(m^3)
V_{mc}	main chamber volume	(m^3)
β_s	conical seat angle	(rad)
H_{max}	maximum shutter ride	(m)
S_{inj}	injectors flow section	(m^2)
$c_{d,inj}$	injectors flow coefficient	
$c_{d,mc}$	main chamber flow coefficient	
k_s	spring constant	(N/m)
F_{so}	spring preload	(N)
$\bar{p}_t$	tank working pressures	(Pa)
$\bar{p}_{cc}$	control chamber working pressure	(Pa)
$\bar{p}_{mc}$	rail and main chamber working pressures	(Pa)
ΔP	av. pressure gradient across piston surfaces	(Pa)
$\overline{ET}$	injectors working Exciting Time	(s)
ρ_{steel}	steel density	(kg/m^3)
μ_{oil}	lubricating oil dynamic viscosity	(kg/(m s))
C_D	drag coefficient	(m/s)
μ_{gas}	gas dynamic viscosity	(kg/(m s))
R	gas constant	(J/(kg K))
T	working temperature	(K)
$\bar{v}$	maximum piston speed	(m/s)
d_s	seat diameter	(m)
ε_p	clearance between piston and cylinder	(m)
$\bar{\delta}$	minimum damping ratio allowed	

Finally, starting from a nominal set of design values, denoted by $x_{i,proj}$, $i = 1,\ldots,8$, we can ask each decision variable to stay within the percentage range $[-\alpha_x\%, +\alpha_x\%]$, where α_x is a parameter whose value needs to be fixed by the user before the optimization process begins; these conditions are expressed through the

Table 5. Problem Formulation: Constants

$k_1 = \frac{\pi}{2}\,\Delta P$	$k_6 = F_{so} - k_s\,H_{max}$
$k_2 = 2\pi\,\mu_{oil}\,\bar{v}$	$k_7 = \frac{\pi}{4}\,\rho_{steel}\,g$
$k_3 = \frac{\pi}{8\mu_{oil}}\,\Delta P$	$a = S_{inj}\,c_{d,inj}$
$k_4 = \frac{\pi}{2}\,\bar{v}$	$\rho_{gas} = \frac{\bar{p}_{cc}}{R\,T}$
$k_5 = \pi\,\Delta P$	$b_1 = 4b_2\,cos^2\beta_s$
$b_2 = \frac{\pi^2 c_{d,mc}\sin\beta_s\cos\beta_s}{4k_s}$	

constraints $h_1(\mathbf{x}),\ldots,h_{16}(\mathbf{x})$. We can notice that the non-negativity constraint directly follows from these last constraints, so that it will always be respected by every solution belonging to the feasible region defined by the other constraints. Also, a similar constraint has to be imposed to the piston thickness, i.e. $(2\,r_p - D_i)$ and $(h_p - h_i)$ are required to be in the range $[-\alpha_x\%, +\alpha_x\%]$ with respect to the corresponding nominal design values, thus obtaining the constraints $g_7(\mathbf{x}),\ldots,g_{10}(\mathbf{x})$.

4.4 The Setting of the Algorithm

In this section we describe the algorithm's setting, starting from the parameters that should be fixed in the construction of the surrogate model. When the optimization process uses surrogate models, some preliminary activities involve the Design of Experiments Tool, the Meta-model Constructor and the external computer simulation tools. The construction of the surrogate model requires a set of experimental data as input. In order to obtain it, we need a preliminary sampling of the design space, which has been conducted using some techniques of Design of Experiments. In particular, we have decided to adopt one strategy generally used in Response Surface Modeling, which is the Central Composite Design (CCD), and, more precisely, the Central Composite Inscribed (CCI) type. This choice is supported by the fact that the design space is quite regular, as the decision variables (factors) can vary between lower and upper boundary values, thus defining an hypercube. The CCI design consists of selecting the *cube points* lying in the interior of the hypercube, *star points* taking minimum and maximum values and *center points* at the origin: so CCI design has five levels per factor. This design has been carried out using the specific functions of the MATLAB's Statistic Toolbox [29].

Once the computer simulations (using external specific tools) have been conducted on the set of designed experiments, the Meta-model Constructor can determine the approximation model, using the kriging technique . To do this, it is necessary to specify the regression model and the correlation model: we choose the linear regression function and the Gaussian correlation function; in addition, we choose the following starting point and bounds for the unknown parameter θ in the correlation model: $\theta_0 = 10$, $lowbd = 10^{-1}$, $upbd = 20$.

Before running the multiobjective optimization algorithm (eventually including a surrogate model among its fitness functions), there are some parameters to be set adequately. Particularly, a preliminary tuning procedure has led to the following choices: the population is composed of 50 individuals, 2 individuals are selected through elitism. Crossover and mutation probabilities have been chosen such that their sum is equal to 1: the crossover fraction has been fixed to 0.7 and consequently the mutation fraction has been set to 0.3.

The size of the archive first has been set equal to 500 elements, which is the default value; it may seem to be a large value, but it is justified by the purpose to reduce the risk to loose some non-dominated solutions (i.e. when the archive is often full during the optimization process). Moreover, while the population is evolving, there could be a great number of non-dominated individuals filling the archive and causing it to move to saturation, even though they can be successively removed by some other solutions entering the archive and dominating them. Finally, it should be preferable to choose an archive having a great capacity, even if it is not fully exploited, especially because it does not determine any additional computational costs, since the empty locations are not involved in the processing of the archive. Then, after 10000 generations, a second phase of the algorithm begins, mainly focussing on densely filling the Pareto front obtained up to that moment; for this reason the maximum size of the archive has been increased up to 1500 elements, presuming to have to face a number of non-dominated individuals greater than that of the previous phase, thus obtaining more crowded fronts. Moreover, it is necessary to specify how many non-dominated feasible solutions have to be injected from the archive into the current population and how often the injection must take place: we choose to inject the oldest individual contained in the archive every 100 generations. The refresh frequency of the archive also need to be established: the archive is updated after each iteration by default and we left this value untouched, but a different choice is possible, that can be influenced by computational considerations: in fact, if the problem to deal with is very complex, the population is large and the archive is completely full, the domination analysis and the update of the archive could be very expensive. Such a situation tends to be avoided, suggesting a low value of the update frequency, allowing a fast evolution of the population, even though the computational burden for the archive update may increase.

Table 6 summarizes the setting of the main parameters of the optimization process.

4.5 Computational Results

This section reports on the computational experiments conducted to illustrate the application of the proposed algorithmic framework on the described case study. In particular, we concentrate the presentation on the basic aspects related to the introduction of the use of the meta-models. At this aim, different sets of experiments have been organized: the first group considering the multiobjective optimization of the functions f_1–f_7 described before; in the second group three functions have been

Table 6. The setting of the multiobjective optimization algorithm

Population	size = 50		
Selection operator	Stochastic Uniform	Elitism = 2	
Crossover operator	Scattered	Probability = 0.7	
Mutation operator	Gaussian	Probability = 0.3	
Archive	$S = 500, 1500$	$f_{upd} = 1$	
Injection process	$f_{inj} = 100$	$n_{inj} = 1$	
Surrogate model	Gaussian Correlation	Linear Regression	$f_N = 1000$

added to the objectives, in order to take account of dynamic aspects of the system in the optimization model.

These new fitness functions, denoted by δp_{mc}, $\sigma(p_{mc})$ and $l.s.(p_{mc})$ as will be discussed further in this subsection, are very expensive to obtain and the optimization model uses their approximations on the basis of an accurate but even time consuming simulation model, being three orders of magnitude between the time required by every single simulation's run and the time necessary for a complete iteration of the EA using surrogates. For the first set of experiments, the project team collaborates to propose a first (tentative) technical solution that we call *nominal* design. So, the optimization algorithm starts with a random population, seeding the nominal element into it, and it runs for 10000 iterations. Then, the solutions contained in the archive are examined using the tools provided by the Solutions Analyzer. In particular, i) different metrics can be used to evaluate the progress of the optimization process and the quality of the achieved front; ii) a clustering analysis can be conducted in order to recognize robust design solutions or individuate crowded regions of the front, as explained in the Section 3.3; iii) the experts opinions on the current solutions can be collected to guide the setting of the algorithm for a new optimization phase. In the second phase the algorithm runs for 5000 more iterations, this time starting from a random selection of 50 archived solutions produced in the previous phase. It is the default option, when the Solutions Archive contains insufficient elements, the population is filled randomly with elements of the last population occurred in the previous phase. More in general, the initial population is composed of a selection of non-dominated solutions obtained at the end of the first phase with a possible integration of new solutions suggested by the analysis of the archive conducted also with the supervision of an expert. The maximum size for the archive switches to 1500 elements and the goal of this phase is further exploring the search space, trying to increase the set of non-dominated solutions.

Intermediate results of the optimization algorithm after 3000 generations consists of a non-dominated front of 61 elements, whose characteristics are reported in Tables 7–8. In these Tables, nominal values, minimum and maximum values as well as average values and variance achieved in the non-dominated front, obtained by

Table 7. Intermediate optimization results without surrogates – Decision Variables after 3000 generations

	Nominal	**Min**	**Max**	**Avg**	**Variance**
x_1	12	12,7	12,7	12,7	0
x_2	30	33	33	33	0
x_3	0,031	0,028	0,033	0,031	$2,9 \times 10^{-6}$
x_4	64	57,6	57,6	57,6	0
x_5	5,5	4,95	6,05	5,52	0,11
x_6	15	15,87	15,87	15,87	0
x_7	43	47,3	47,3	47,3	0
x_8	16,5	15,97	15,97	15,97	0

Table 8. Intermediate optimization results without surrogates – Objective Functions after 3000 generations

	Nominal	**Min**	**Max**	**Avg**	**Variance**
f_1	7,40	4,04	4,04	4,04	0
f_2	59,89	4,17	4,17	4,17	0
f_3	312,07	317,02	374,82	338,38	395,62
f_4	27,62	23,76	29,04	26,12	2,56
f_5	$0,48 \times 10^5$	$0,36 \times 10^5$	$0,43 \times 10^5$	$0,39 \times 10^5$	$4,3 \times 10^6$
f_6	6,17	7,6	7,6	7,6	0
f_7	3,90	4,35	4,35	4,35	0

the optimization process, are reported for each decision variable and each objective function, respectively.

At the end of this group of experiments, i.e. after 15000 generations, the design optimization process produces a front of non-dominated solutions containing 134 elements, whose characteristics are reported in Tables 9–10. These results satisfy all mechanical and control requirements incorporated in the optimization model. As it can be seen from the variance of the solutions, which is almost zero considering both variables and fitnesses, the algorithm tends to converge, and the only variables whose values still vary are x_3, x_4 and x_5.

The next step consists of a more accurate analysis in which a detailed simulation model of the system is employed to obtain an evaluation of the dynamic performance of each current solution. This evaluation allows the designers to rank the candidate

Table 9. Optimization results without surrogates – Decision Variables after 15000 generations

	Nominal	**Min**	**Max**	**Avg**	**Variance**
x_1	12	12,75	12,75	12,75	0
x_2	30	33	33	33	0
x_3	0,031	0,028	0,034	0,033	$5,1 \times 10^{-6}$
x_4	64	57,6	57,72	57,6	$1,1 \times 10^{-4}$
x_5	5,5	4,95	5,90	5,12	0.036
x_6	15	15,93	15,94	15,94	~ 0
x_7	43	47,3	47,3	47,3	0
x_8	16,5	16,04	16,04	16,04	0

Table 10. Optimization results without surrogates – Objective Functions after 15000 generations

	Nominal	**Min**	**Max**	**Avg**	**Variance**
f_1	7,40	4,06	4,06	4,06	0
f_2	59,89	4,18	4,18	4,18	0
f_3	312,07	308,38	376,06	320,02	619,84
f_4	27,62	24,38	29,04	28,1	0,97
f_5	$0,48 \times 10^5$	$0,36 \times 10^5$	$0,41 \times 10^5$	$0,36 \times 10^5$	$5,9 \times 10^5$
f_6	6,17	7,66	7,66	7,66	0
f_7	3,90	4,37	4,37	4,37	0

designs. In general, this operation can be repeated cyclically during the optimization process affecting the evolution mechanism of the algorithm only if it requires a limited increase in the computational effort. In our first experimental study, since the simulation analysis is quite expensive, it has been performed once, at the end of the evolutionary optimization process; the results are described in what follows.

The accurate simulation model, proposed in [23], is used to analyze the dynamic of the current solutions. The simulation tool allows the project team to express new preference criteria not considered yet in the optimization model, and not suitable to be incorporated in the model straightforward. These new preference criteria are based on the following considerations.

First of all, we consider that, assuming the same inputs, the working pressure changes with the system geometrical configuration. As a consequence, the same step variation of the solenoid valve driving signal causes a different steady state pressure

change. A wide pressure change implies a less accurate control, because the control action sensibly affect the controlled variable, even in closed loop conditions. On the contrary, small changes restrict the working pressure range, thus reducing efficiency during high accelerations, different loads or idle speed conditions. To achieve good performances both in steady state and transient conditions, a trade-off among these solutions is necessary. In this framework, we suppose that a small pressure change δp_{mc} $[Pa]$ is desirable. The system equilibrium points can be computed directly from the model differential equations. Nevertheless, due to input discontinuities and model nonlinearities and singularities, we achieve the same result in a simpler way by integrating the model equations, considering different valve driving signal values within a fixed range.

A second criterion for the design selection is based on the reduction of control and rail pressures oscillating behaviour within the control period. In fact, the pressure increases when the solenoid valve is opened, while decreases when it is closed, and these pressure variations affect the injection accuracy. Since their amplitude cannot be easily predicted from static computations, we can estimate it by simulating the dynamic model, and by considering the injection pressure standard deviation $\sigma(p_{mc})$ $[Pa]$ with respect to the average value during steady state conditions.

Finally, a third parameter evaluates the influence of speed and injection duration on the rail pressure dynamics. To this end, starting from a steady state condition, we simulate load and engine speed (and thus injection timings) step variations, while keeping the valve driving signal constant. Then we calculate the least square of the difference between the actual rail pressure and the initial steady state pressure $l.s.(p_{mc})$ $[Pa^2]$, considering that lower values correspond to better performances.

Figures 8 and 9 depict the values assumed by the performance indices of the solutions coming from the optimization process (the first one being the nominal individual) after 3000 and 15000 iterations, respectively. Each index has been normalized with respect to its minimal value, thus the best individual has a performance index equal to 1. Clearly, according to this criterion, the absolute preferred individual by the designer team is the one minimizing all the indices at the same time. It comes from Fig. 8 that the optimization procedure after 3000 iterations does not produce satisfactory results, both due to the small number of non-dominated solutions and to the difficulty in making a choice considering the corresponding dynamical indices values. Moreover, a noticeable variability of the dynamical indices implies a lower number of technically suitable solutions in the set of the non-dominated points collected in the Solutions Archive. These observations and the expert's agreement conduct to consider current results as quite preliminary in the evolutionary optimization process.

After 15000 iterations (Fig. 9), it is possible to identify three individuals with performance indices close to 1 (♯31, ♯39, ♯41), the best being the ♯39, which has the following components and objectives values:

$$\bar{\mathbf{x}} = [12.748, 33, 0.033, 57.6, 5.9, 15.94, 47.3, 16.04] \ ,$$

$$\bar{\mathbf{f}} = \left[4.06, 4.18, 308.7, 24.38, 0.41 \times 10^5, 7.66, 4.37\right] \ .$$

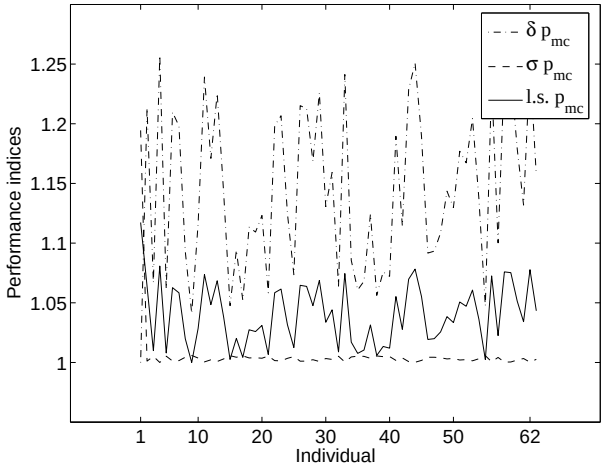

Fig. 8. Performance indices for the individuals coming from the optimization process (3000 generations without surrogates)

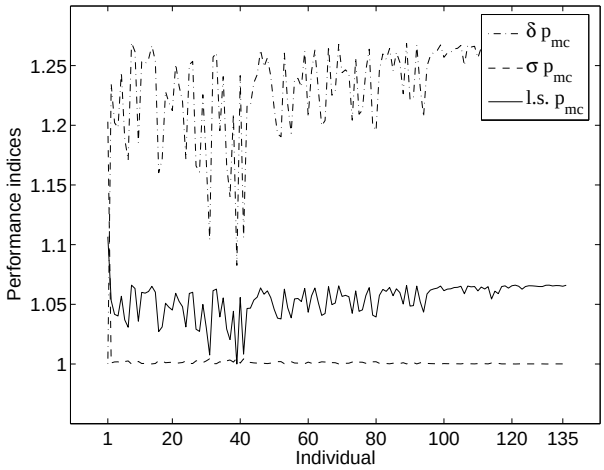

Fig. 9. Performance indices for the individuals coming from the optimization process (15000 generations without surrogates)

The vector of the adjunctive indices $[\delta p_{mc}, \sigma(p_{mc}), l.s.(p_{mc})]$ attained by the selected solution ♯39 is: $\left[1.097 \times 10^6, 0.083, 0.145 \times 10^{10}\right]$.

As previously explained, the optimization model employed in the experiments just described does not include any fitness functions evaluating the dynamic behaviour of the system. This kind of evaluations is conducted only at the end of the optimization process using (off-line) specific simulation models. This choice is due to the high computational effort which would have been requested if they were incorporated in the optimization model. Nevertheless, it is possible to take account of

them in the optimization model considering their approximations, using the kriging technique as discussed early in the Chapter.

In this case the optimization process evolves in an iterative way, as explained in the following: using a DOE technique, based on Central Composite Design, we produce a number of experimental data that are evaluated through accurate simulation studies, in order to develop the surrogate models to be employed in the optimization. Once the surrogates are generated, they are inserted as adjunctive objective functions in the optimization model and the algorithm starts, using the same initial population as before.

Since the starting sampling points for the model construction can be even far from the region in which the solutions of the optimization process are concentrating, it is reasonable to update the surrogate models every f_N iterations (i.e. adopting a generation-based evolution control). The sampling points that need to be integrated in the surrogate model are those contained in the archive, i.e. the non-dominated solutions found up to that moment; doing so we fill the surrogates according to the optimization results, making the models more accurate in the most promising areas. So the optimization process restarts, using the updated surrogates.

At the end of the optimization procedure, a front of non-dominated elements is obtained, containing 251 points; their characteristics are reported in Tables 11–12, according to the scheme followed for the results of the previous experiment.

Table 11. Optimization results using surrogates – Decision Variables after 3000 generations

	Nominal	**Min**	**Max**	**Avg**	**Variance**
x_1	12	12,76	12,76	12,76	0
x_2	30	33	33	33	0
x_3	0,031	0,028	0,033	0,031	$3,4 \times 10^{-6}$
x_4	64	60,41	61,13	60,88	0,016
x_5	5,5	4,95	5,65	5,14	0,011
x_6	15	15,95	15,95	15,95	0
x_7	43	47,3	47,3	47,3	0
x_8	16,5	16,05	16,05	16,05	0

The solutions obtained are evaluated according to the dynamic performance indices already described, whose values are plotted in Fig. 10.

Note that there are three solutions all minimizing the indices (♯41, ♯43, ♯47), so – even though the number of solutions among which choosing the preferrable one has been extremely reduced – the final decision will be taken also relying upon human expertise; in case of lots of solutions (e.g. more than 10) the designer might run other experiments, appropriately designed to rank non-dominated solutions, or he

Table 12. Optimization results using surrogates – Objective Functions after 3000 generations

	Nominal	**Min**	**Max**	**Avg**	**Variance**
f_1	$7,40$	$4,07$	$4,07$	$4,07$	0
f_2	$59,89$	$4,18$	$4,18$	$4,18$	0
f_3	$312,07$	$315,35$	$376,24$	$339,81$	$461,46$
f_4	$27,62$	$26,19$	$29,94$	$28,78$	$0,31$
f_5	$0,48 \times 10^5$	$0,36 \times 10^5$	$0,39 \times 10^5$	$0,36 \times 10^5$	$1,38 \times 10^5$
f_6	$6,17$	$7,67$	$7,67$	$7,67$	0
f_7	$3,90$	$4,37$	$4,37$	$4,37$	0

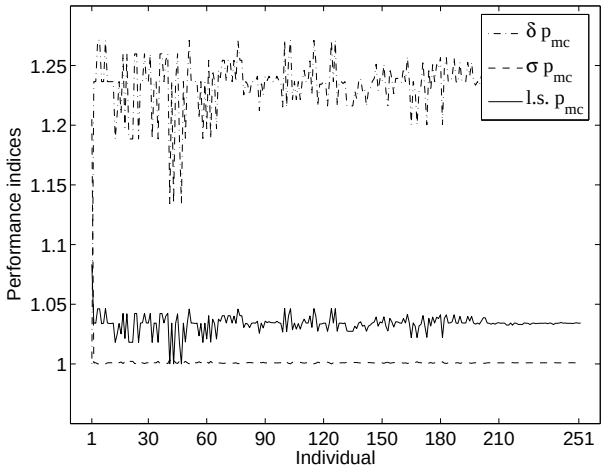

Fig. 10. Performance indices for the individuals coming from the optimization process (3000 generations using surrogates)

might adopt some adequate multi-criteria analysis tool. Tables 13–14 report on the values of the decision variables and the objective functions, respectively, for the best individuals obtained at the end of the optimization process, compared with those of the nominal individual (♯1).

Comparing the results obtained at the end of this experiment with those obtained in the previous one, it is clear the advantage offered by the use of the surrogates. Even if a lower number of generations is performed (15000 in the first case and only 3000 in the second), in the latter case the analysis conducted at the end of the process shows a higher number and a general better quality of non-dominated solutions. In particular, the optimization process using surrogates produces 251 individuals (see Fig. 10), while only 134 individuals come from the optimization without surrogates after 15000 iterations (see Fig. 9). The difference is much more evident by compar-

Table 13. Final optimization results using surrogates – Decision Variables after 3000 generations

	♯**1**	♯**41**	♯**43**	♯**47**
x_1	12	12,76	12,76	12,76
x_2	30	33	33	33
x_3	0,031	0,032	0,032	0,032
x_4	64	60,98	60,91	60,86
x_5	5,5	5,65	5,65	5,65
x_6	15	15,95	15,95	15,95
x_7	43	47,3	47,3	47,3
x_8	16,5	16,05	16,05	16,05

Table 14. Final optimization results using surrogates – Objective Functions after 3000 generations

	♯**1**	♯**41**	♯**43**	♯**47**
f_1	7,40	4,07	4,07	4,07
f_2	59,89	4,18	4,18	4,18
f_3	312,07	327,17	327,17	327,17
f_4	27,62	26,22	26,20	26,19
f_5	$0,48 \times 10^5$	$0,39 \times 10^5$	$0,39 \times 10^5$	$0,39 \times 10^5$
f_6	6,17	7,67	7,67	7,67
f_7	3,90	4,37	4,37	4,37
δp_{mc}	$8,69 \times 10^5$	$9,86 \times 10^5$	$9,86 \times 10^5$	$9,86 \times 10^5$
σp_{mc}	0,099	0,083	0,083	0,083
$l.s.p_{mc}$	$0,161 \times 10^{10}$	$0,148 \times 10^{10}$	$0,148 \times 10^{10}$	$0,148 \times 10^{10}$

ing these results with those obtained from the optimization process without surrogates after 3000 iterations, which gives only 61 individuals (see Fig. 8). It must be also mentioned that the individuals obtained in this case are almost all dominated by those obtained in the two others, and they result strictly dominated if the new indices are considered. Clearly, the optimization process using surrogates gives a better percentage of solutions providing high dynamic performances and a reduced variability in the indices related to the dynamic response of the system, as can be deduced from Table 15. This Table shows the normalized standard deviation (with respect to the

Table 15. Normalized standard deviations of the dynamical performance indices depending on the optimization method

	Surr. 3000 gen.	**no Surr. 3000 gen.**	**no Surr. 15000 gen.**
δp_{mc}	1.00	3.28	1.84
$\sigma(p_{mc})$	1.00	3.48	1.86
$l.s.(p_{mc})$	1.00	3.47	1.90

lower value) of the dynamical performance indices, depending on the optimization method, where the lower standard deviation for each index means a higher number of suitable solutions.

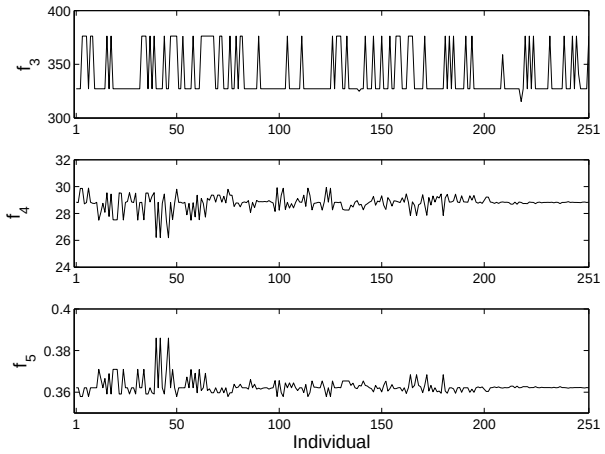

Fig. 11. Objectives f_3, f_4 and f_5 in the optimization process using surrogates

Some remarks can be pointed out by analyzing the static and dynamical performance objectives resulting from the optimization process including surrogates, by taking into account the expert's opinion summarized in what follows. To this aim, Fig. 11 depicts for each individual the objectives having a variance greater than zero (according to Table 12), i.e. f_3, f_4 and f_5, respectively.

It should be noticed that the objective function f_3 depends on three decision variables, namely x_1, x_2, x_3, but only x_3 has a variance greater than zero. Since x_3 varies only among three different values, it causes f_3 to do the same. These three values of x_3 are those that guarantee a good system performance so that the algorithm almost fixes these values, to let the optimization proceed.

Another consideration comes by comparing f_4 and $l.s.(p_{mc})$, which have analogous trends. This indicates that both the objectives aim to similar targets, thus it would be possible to include the information carried by $l.s.(p_{mc})$ into f_4.

Moreover, although f_4 and f_5 do not entirely depend on the same parameters, their trend is specular and conflicting. This is particularly evident for the final best three individuals (♯41, ♯43, ♯47) according to the dynamical indices. The objective function f_4, which represents the main circuit time constant, primarily depends on the rail volume, while f_5, which almost takes into account the final pressure value in presence of injections, is greatly affected by the flow sections. It is worth to note that all the three best individuals have high values of the objective f_5, which could affect the accurate metering of fuel during injections, which is a crucial feature for the engine performances and pollutant emissions. In fact, f_5 takes into account the influence of a single injection on the rail pressure within an injection cycle. Since it is not possible to individually regulate the injection timing for each injector, it is difficult for the controller to compensate the pressure drop between two consecutive injections. If these issues are taken into account, the alternative solutions deriving from the optimization process become more significant, balancing different requirements at the same time. Also, from this analysis it comes that an improvement of performances could be achieved through a proper tuning of the injectors geometry, which has not been included in the optimization model.

As for the other dynamical performance indices, δp_{mc} gives helpful information on the whole system dynamics, as it is affected by both the main circuit time constant and the control circuit time constant, the latter depending also on the control valve geometry. Finally, the small variability of the objective $\sigma(p_{mc})$ is due to the small variance of the parameters affecting it. About this argument, it is important to note that the discussion devoted to illustrate the application of the proposed framework on a real case, is based, for sake of simplicity, on a single run of the procedure on the proposed experimental tests. In practice, due to the intrinsic random behaviour of some components of the proposed approach, a whole campaign of computational experiments has to be conducted on each design task endig the process with an accurate statistic elabolation of the results.

It must also be recognized by the expert that almost all conclusions and observations can only be deduced by analyzing the optimization results offering interesting ways to significantly improve the iterative MDO process. A promising direction, at this aim, may consist of fixing the design variables that present a null variance at the end of the optimization process. This practice reduces the size of the problem and allows the project team to concentrate the efforts on the other relevant aspects going toward the design convergence.

5 Conclusion

The flexibility of evolutionary methods to handle different (and complex) objectives joint to their easier possible extension to multicriteria optimization, makes these techniques suitable to play a relevant role as components of decision support tools for the design of complex systems.

This Chapter describes an optimization software framework devoted to MDO activities in the control domain. Moreover, it illustrates the application of an en-

hanced evolutionary optimization method in the context of a real mechanical design. Namely, the optimal design of an automotive injection system is discussed. Computational experiments, conducted on a case study, showed the optimization ability and the effectiveness of the proposed approach. These results highlight the contribute of the use of enhanced algorithmic schemes in which optimization solvers are assisted by surrogate models and suggest different promising directions for further researches.

Further developments and improvements of the software framework will be dedicated to include other multiobjective optimization algorithms. These algorithms should be used either as alternative to the previous genetic-based algorithm or they should be composed in a hybrid optimization scheme [3]. Recently, a large part of researchers interest, in the MDO context, has been captured by particle swarm optimization (PSO) methods. Since this technique seems to be suitable both to operate with surrogates and to integrate other optimization approaches, it appears as a good candidate to be included in the optimization framework. Another promising direction to improve the behaviour of the multiobjective optimizer relies with the use of local search techniques to hybridize its search mechanism. At this aim a fast single-objective search can be adopted optimizing an aggregate objective function [5, 30, 32], while different strategies can be pursued in order to apply an improving pressure either on the current population or the current archive.

One drawback in the proposed optimization scheme is that the frequency of evolution control is fixed. This is not very practical because the fidelity of the approximation model may vary significantly during optimization. In fact, a predefined evolution control frequency may cause strong oscillation during optimization due to large model errors. It is straightforward to imagine that the frequency of evolution control should depend on the fidelity of the approximation model. A strategy to adjust the frequency of evolution control can be associated to the generation-based approach. Another useful improvement may refer to the introduction of some multicriteria analysis tool to assist the human esperts in the selection of the design candidates proposed by the optimization process.

6 ACKNOWLEDGMENTS

The authors wish to thank two anonymous referees for their valuable comments and criticisms which have contributed to improve the quality of the presentation of their work.

References

1. Alexandrov N.M., Hussaini M.Y. (Eds) (1997) Multidisciplinary Design Optimization - state of the art. Proceedings of the ICASE/NASA Langley Workshop on Multidisciplinary Design Optimization, SIAM Proceedings Series.

2. Amorese C., De Matthaeis S., De Michele O. and Satriano A. (2004) The Gaseous Fuel Option: LPG and CNG. In: Proceedings of the International Conference on Vehicles Alternative Fuel System & Environmental Protection, Dublin, Ireland.
3. Burke M.E. and Kendall G. (2005) Search Methodologies. Springer, New York.
4. Deb K. (1999) Evolutionary Algorithms for MultiCriterion Optimization in Engineering Design. In: Miettinen K., Mäkelä M., Neittaanmäki P., and Périaux (Eds) Proceedings of Evolutionary Algorithms in Engineering and Computer Science (EUROGEN'99), Jyväskylä, Finland, pp. 135–161.
5. Deb K. (2001) Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons.
6. Deb K. (2004) Optimization for Engineering Design: Algorithms and Examples. Prentice-Hall.
7. Deb K., Pratap A., Agarwal S. and Meyarivan T. (2002) A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2): 182–197.
8. Ehrgott M. (2005) Multicriteria Optimization. Springer-Verlag, Berlin.
9. Franklin G.F., Powell J.D., and Emami-Naeini A. (2002) Feedback Control of Dynamic Systems. Prentice Hall, Upper Saddler River NJ.
10. Guzzella L. and Amstutz A. (1998) Control of Diesel Engines. IEEE Control Systems Magazine, 18(5): 53–71.
11. Guzzella L. and Honder C.H. (2004) Introduction to Modeling and Control of Internal Combustion Engine Systems. Springer-Verlag, Berlin - Heidelberg.
12. Wu C.F.J. and Hamada M. (2000) Experiments: Planning, Analysis and Parameter Design Optimization. John Wiley & Sons.
13. Jin Y. (2002) Knowledge in Evolutionary and Learning Systems. Shaker, Aachen.
14. Jin Y. (2005) A Comprehensive Survey of Fitness Approximation in Evolutionary Computation. Soft Computing, 9(1): 3–12.
15. Jin Y., Chen W., and Simpson T.W. (2001) Comparative Studies of Metamodeling Techniques under Multiple Modeling Criteria. Structural and Multidisciplinary Optimization, 23(1): 1–13.
16. Jin Y., Olhofer M., and Sendhoff B. (2000) On Evolutionary Optimization with Approximate Fitness Functions. In: Proceedings of the Genetic and Evolutionary Computation Conference, Las Vegas, Nevada. Morgan Kaufmann, pp. 786-793.
17. Jin Y., Olhofer M., and Sendhoff B. (2001) Managing Approximate Models in Evolutionary Aerodynamic Design Optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, 1: 592-599.
18. Jin Y., Olhofer M., and Sendhoff B (2002) A Framework for Evolutionary Optimization with Approximate Fitness Functions. IEEE Transactions on Evolutionary Computation, 6(5): 481-494.
19. Jin Y. and Sendhoff B. (1999) Knowledge Incorporation into Neural Networks from Fuzzy Rules. Neural Processing Letters, 10(3): 231-242.
20. Joshi S.S. (1999) The Need for a Systems Perspective in Control Theory and Practice. IEEE Control Systems Magazine, 19(6): 56–63.
21. Kodiyalam S. and Sobieszczanski-Sobieski J. (2001) Multidisciplinary Design Optimization – Some Formal Methods, Framework Requirements, and Application to Vehicle Design. International Journal of Vehicle Design, 25(1/2): 3-22.
22. Lino P. (2004) Problemi di Modellistica di Sistemi di Iniezione Innovativi per Motori a Combustione Interna. Ph.D. Thesis (in italian), Università di Catania.

23. Lino P., Maione B., Amorese C. and De Matthaeis S. (2006) Modeling and Predictive Control of a New Injection System for Compressed Natural Gas Engines. In: Proceedings of the IEEE CCA 2006 International Conference, Munich, Germany.
24. Lino P., Maione B., and Rizzo A. (2005) A Control-Oriented Model of a Common Rail Injection System for Diesel Engines. In: Proceedings of the International IEEE ETFA Conference, Catania, Italy, vol. 1, pp. 557–563.
25. Lophaven S.N., Nielsen H.B., Søndergaard J. (2002) DACE — A MATLAB Kriging Toolbox. Version 2.0. Technical Report IMM-TR-2002-12
26. The MathWorks Inc. (2003) Neural Networks Toolbox, Natick, Massachussetts.
27. The MathWorks Inc. (2004) Genetic Algorithm and Direct Search Toolbox, Natick, Massachussetts.
28. The MathWorks Inc. (2005) Optimization Toolbox, Natick, Massachussetts.
29. The MathWorks Inc. (2005) Statistics Toolbox, Natick, Massachussetts.
30. Messac A., Sundararaj G.J., Tappeta R.V., and Renaud J.E. (2000) Ability of Objective Functions to Generate Points on Non-Convex Pareto Frontiers. American Institute of Aeronautics and Astronautics Journal, 38(6): 1084–1091.
31. Michalewicz Z., Dasgupta D., Le Riche R.G., and Schoenauer M. (1996) Evolutionary Algorithms for Constrained Engineering Problems, Computers & Industrial Engineering Journal, 30(2): 851–870.
32. Miettinen K.M. (1999) Nonlinear Multiobjective Optimization. Kluwer Academic Publishers.
33. Montgomery D.C. (2005) Design and Analysis of Experiments. John Wiley & Sons.
34. Myers R.H. and Montgomery D.C. (1995) Response Surface Methodology: Process and Product Optimization Using Designing Experiments. John Wiley & Sons.
35. Nain P.K.S. and Deb K. (2005) A Multi-Objective Optimization Procedure with Successive Approximate Models. KanGAL Report No. 2005002.
36. Okabe T., Jin Y. and Sendhoff B. (2003) A Critical Survey of Performance Indices for Multi-Objective Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, 2: 878–885.
37. Ong Y.S., Nair P.B., Keane A.J. (2003) Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling. American Institute of Aeronautics and Astronautics Journal, 41(4): 687–696.
38. Osyczka A. (2002) Evolutionary Algorithms for Single and Multicriteria Design Optimization. Studies in Fuzzyness and Soft Computing, Physica-Verlag, Heidelberg.
39. Papalambros P.Y. and Wilde D.J. (2000) Principles of Optimal Design. Cambridge University Press.
40. Peña D. and Prieto F.J. (2001) Cluster Identification Using Projections. Journal of the American Statistical Association, 96(456): 1433-1445.
41. Sobieszczanski-Sobieski J. (1995) Multidisciplinary Design Optimization: an Emerging New Engineering Discipline. In: Herskovits J. (Ed.) Advances in Structural Optimization. Kluwer A.P., Dordrecht, pp. 483-496.
42. Streeter V., Wylie K. and Bedford E. (1998) Fluid Mechanics. 9th ed McGraw-Hill, New York.
43. Ulrich K.T. and Eppinger S.D. (1995) Product Design and Development. McGraw-Hill.
44. Van Veldhuizen D.A. and Lamont G.B. (2000) On Measuring Multiobjective Evolutionary Algorithm Pperformance. In: Proceedings of the 2000 IEEE Congress on Evolutionary Computation, pp. 204–211.
45. Vanderplaats P. (2001) Numerical Optimization Techniques for Engineering Design, VR&D.

46. Wijetunge R.S., Brace C.J., Hawley J.G., Vaughan N.D., Horrocks R.W. and Bird G.L. (1999) Dynamic Behaviour of a High Speed Direct Injection Diesel Engine. SAE Technical Paper, 1999-01-0829
47. Zitzler E., Deb K. and Thiele L. (2000) Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation, 8(2): 173–195.
48. Zucrow M. and Hoffman J. (1976) Gas Dynamics. John Wiley & Sons, New York.